PORTING ESP32FORTH TO THE T-DECK

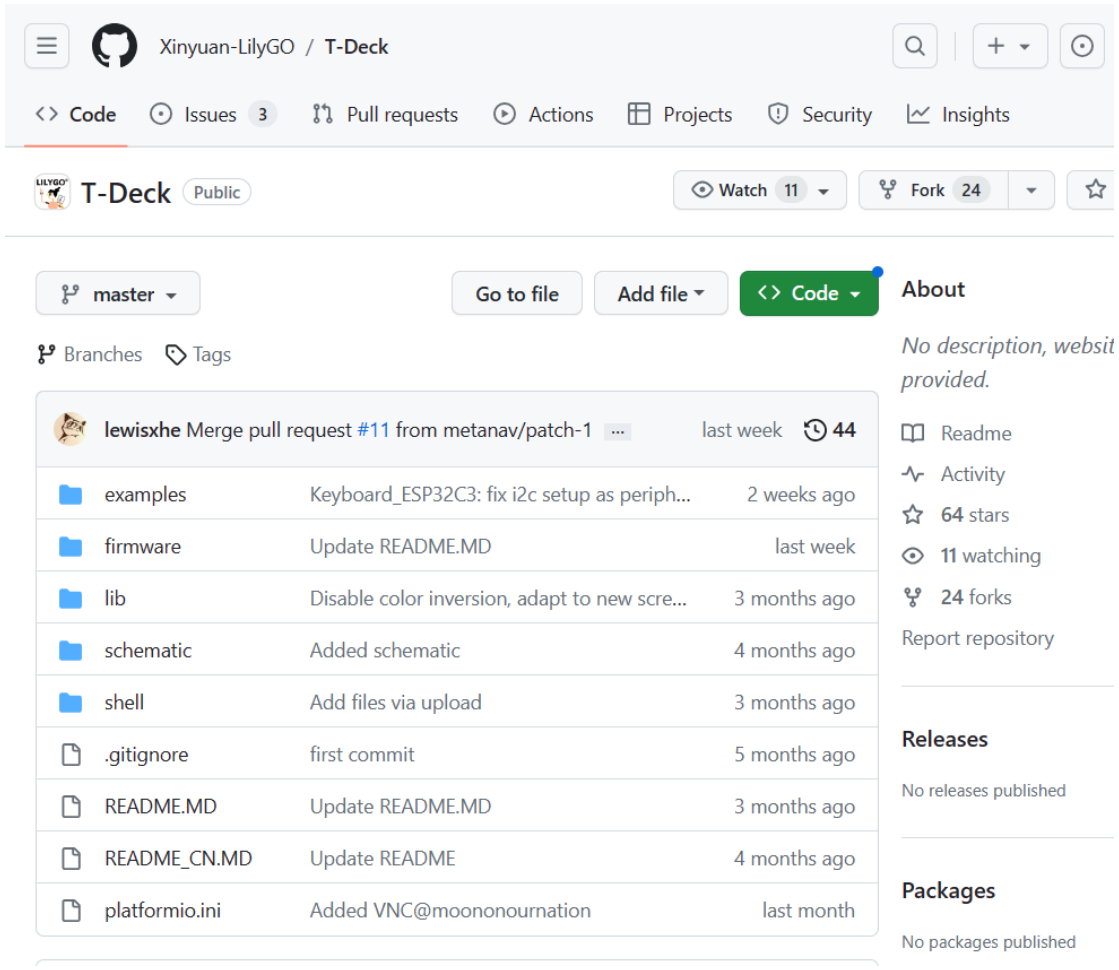
JASON CJ TAY

WHAT IS THE LILYGO T-DECK

- ESP32-S3 based portable device
- Reminiscent of a Blackberry
- In fact, it uses a Blackberry keyboard
- Ah, the keyboard… and the trackball
- With or without LoRa
- PSRAM, Micro SD Card, microphone, speaker
- 3D printable casing
- Battery? Try 800mAh Li-Po.

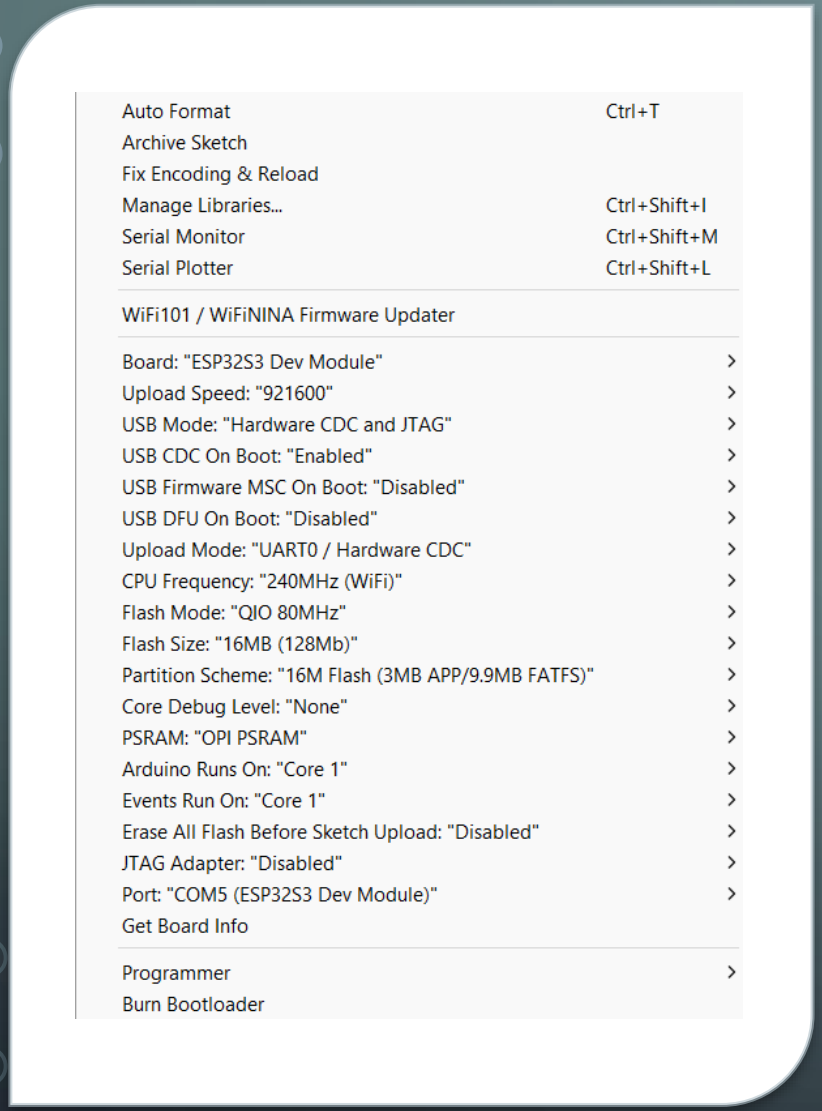
HOW DOES IT COMPARE?

- Better keyboard feel, good for two handed use
- Bigger, faster display
- Not turnkey like M5Stack
- Cheaper
- More hardware for the $$
- No markings for other characters ☹



DRIVERS FROM GITHUB

- Go to LilyGo's GitHub to get the drivers for Arduino
- Xinyuan-LilyGO/T-Deck (github.com)
- Keyboard on I2C
- Display on SPI
- There's actually an ESP32-C3 on board as well



CONFIGURE ARDUINO

- Configure settings as per the image
- Plonk in the libraries
- Get your ESP32forth source
- Make sure it builds successfully
- Get the text buffer display lib from:
- [electricidea/M5StickC-TB_Display: A simple scrolling text display library for the M5StickC (github.com)]

START MODIFYING

- Add in library headers
- Define I2C keyboard slave address (the ESP32-C3), and pin nos.
- Declare the display instance

```
67 // SD_MMC does not work on ESP32-S2 / ESP32-C3
68 #if !defined(CONFIG_IDF_TARGET_ESP32S2) && !defined(CONFIG_IDF_TARGET_ESP32C3)
69 # define ENABLE_SD_MMC_SUPPORT
70 #endif
71
72 // ESP32-C3 has no DACs.
73 #if !defined(CONFIG_IDF_TARGET_ESP32C3) && !defined(CONFIG_IDF_TARGET_ESP32S3)
74 # define ENABLE_DAC_SUPPORT
75 #endif
76
77 #if !defined(CONFIG_IDF_TARGET_ESP32C3)
78 # define ENABLE_SERIAL2_SUPPORT
79 #endif
80
81 // RMT support designed around v2.0.1 toolchain.
82 // While ESP32 also has RMT, for now only include for
83 // ESP32-S2 and ESP32-C3.
84 #if defined(CONFIG_IDF_TARGET_ESP32S2) || \
85     defined(CONFIG_IDF_TARGET_ESP32C3) || \
86     defined(SIM_PRINT_ONLY)
87 # define ENABLE_RMT_SUPPORT
88 #endif
```

ADJUST FEATURES

- Existing code will omit Serial2 support on the S3
- Adjusting the code this way will help support more types of ESP32

```
112 #define VOCABULARY_LIST \
113   V(forth) V(internals) V(tdeck) \
114   V(rtos) V(SPIFFS) V(serial) V(SD) V(SD_MMC) V(ESP) \
115   V(ledc) V(Wire) V(WiFi) V(bluetooth) V(sockets) V(oled) \
116   V(rmt) V(interrupts) V(spi_flash) V(camera) V(timers)
117
```

ADD A VOCABULARY LIST FOR THE T-DECK

ADD IN THE HIGHLIGHTED TERM SO THAT WE CAN ADD IN T-DECK SPECIFIC VOCAB

```
508 #define REQUIRED_SERIAL_SUPPORT \
509   XV(serial, "Serial.begin", SERIAL_BEGIN, Serial.begin(tos); DROP) \
510   XV(serial, "Serial.end", SERIAL_END, Serial.end()) \
511   XV(serial, "Serial.available", SERIAL_AVAILABLE, PUSH Serial.available()) \
512   XV(serial, "Serial.readBytes", SERIAL_READ_BYTES, n0 = Serial.readBytes(b1, n0); NIP) \
513   XV(serial, "Serial.write", SERIAL_WRITE, n0 = Serial.write(b1, n0); NIP) \
514   XV(serial, "Serial.flush", SERIAL_FLUSH, Serial.flush())
515
516 int tdavail() { Wire.requestFrom(LILYGO_KB_SLAVE_ADDRESS, 1); return Wire.available(); }
517
518 int tdgetch()
519 {
520   int keyValue = 0;
521   Wire.requestFrom(LILYGO_KB_SLAVE_ADDRESS, 1);
522   if(Wire.available() > 0) keyValue = Wire.read();
523   return keyValue;
524 }
525
526 void tdwrite(uint8_t *buf, int len)
527 {
528   if(buf != NULL)
529     for(int i=0; i<len; i++) tb_display_print_char(buf[i]);
530 }
531
532 #define REQUIRED_TD_SUPPORT \
533   XV(tdeck, "tdkey", TD_KEY, PUSH tdgetch()) \
534   XV(tdeck, "tdkey?", TD_AVAIL, PUSH tdavail()) \
535   XV(tdeck, "tdwrite", TD_WRITE, tdwrite(b1, n0); DROPn(2))
536
```

ADD IN NEW WRAPPERS

- These new wrapper words will enable us to divert the standard key-in and char-out functions from the serial port to the T-Deck keyboard and display, respectively
- It is convenient to add it in approximately after the existing “REQUIRED_SERIAL_SUPPORT” section

```
454
455 #define PLATFORM_OPCODE_LIST \
456   USER_WORDS \
457   REQUIRED_ESP_SUPPORT \
458   REQUIRED_MEMORY_SUPPORT \
459   REQUIRED_SERIAL_SUPPORT \
460   REQUIRED_ARDUINO_GPIO_SUPPORT \
461   REQUIRED_SYSTEM_SUPPORT \
462   REQUIRED_FILES_SUPPORT \
463   REQUIRED_TD_SUPPORT \
464   OPTIONAL_SERIAL2_SUPPORT \
465   OPTIONAL_LEDC_SUPPORT \
466   OPTIONAL_DAC_SUPPORT \
467   OPTIONAL_SPIFFS_SUPPORT \
468   OPTIONAL_WIFI_SUPPORT \
469   OPTIONAL_MDNS_SUPPORT \
470   OPTIONAL_SD_SUPPORT \
471   OPTIONAL_SD_MMC_SUPPORT \
472   OPTIONAL_I2C_SUPPORT \
473   OPTIONAL_SERIAL_BLUETOOTH_SUPPORT \
474   OPTIONAL_CAMERA_SUPPORT \
475   OPTIONAL_SOCKETS_SUPPORT \
476   OPTIONAL_FREERTOS_SUPPORT \
477   OPTIONAL_INTERRUPTS_SUPPORT \
478   OPTIONAL_RMT_SUPPORT \
479   OPTIONAL_OLED_SUPPORT \
480   OPTIONAL_SPI_FLASH_SUPPORT \
481   FLOATING_POINT_LIST
```

PLATFORM SUPPORT LIST

- Now go back to the Platform Support list and add in our new T-Deck vocabulary
- See in the list, “REQUIRED_TD_SUPPORT”

```
2933 void setup() {
2934   tdEnabledFeatures = TD_FEATURE_NONE;
2935   //!⚠ The board peripheral power control pin needs to be set to HIGH when using the peripheral
2936   pinMode(BOARD_POWERON, OUTPUT);
2937   digitalWrite(BOARD_POWERON, HIGH);
2938 
2939   // There needs to be a delay after power on, give LILYGO-KEYBOARD some startup time
2940   //delay(500);
2941   tft.begin();
2942   tft.setRotation( 1 );
2943   tft.fillScreen(TFT_BLACK);
2944   tft.setTextColor(TFT_WHITE);
2945   delay(200);
2946   Wire.begin(BOARD_I2C_SDA, BOARD_I2C_SCL);
2947 
2948   // Check keyboard
2949   Wire.requestFrom(LILYGO_KB_SLAVE_ADDRESS, 1);
2950   if (Wire.read() != -1) tdEnabledFeatures |= TD_FEATURE_KEYBOARD;
2951   tb_display_init(1);    // Force it always to the same orientation.
2952 
2953   cell_t fh = heap_caps_get_free_size(MALLOC_CAP_INTERNAL);
2954   cell_t hc = heap_caps_get_largest_free_block(MALLOC_CAP_INTERNAL);
2955   if (fh - hc < MINIMUM_FREE_SYSTEM_HEAP) {
2956     hc = fh - MINIMUM_FREE_SYSTEM_HEAP;
2957   }
2958   cell_t *heap = (cell_t *) malloc(hc);
2959   forth_init(0, 0, heap, hc, boot, sizeof(boot));
2960 }
```

ADJUST SETUP()

- It is necessary to adjust the Arduino setup() function now, so that we initialize the display and the keyboard as required
- After this, we need to look for the Forth equivalent of avail, getch and putch bindings

PATCH IN OUR KEYBOARD AND DISPLAY

```
2260 ( Set up Basic I/O )
2261 internals definitions also serial
2262 : esp32-bye   0 terminate ;
2263 : serial-type ( a n -- ) Serial.write drop ;
2264 : serial-key ( -- n )
2265    begin pause Serial.available until 0 >r rp@ 1 Ser
2266 : serial-key? ( -- n ) Serial.available ;
2267 also forth definitions
2268 : default-type tdwrite ;
2269 : default-key tdkey ;
2270 : default-key? tdkey? ;
2271 ' default-type is type
2272 ' default-key is key
2273 ' default-key? is key?
2274 ' esp32-bye is bye
2275 only forth definitions
```

- A large part of the ESP32forth source code is written in Forth
- Search for the Basic I/O definitions for ESP32forth's runtime system
- Cheekily patch-in our I2C keyboard and SPI display routines
- Compile!

WHERE TO FROM HERE?

- Lvgl implementation
  - Better scrolling, trackball use
- Sort out printing of additional characters, and a way to display a legend
  - <, >, =, for starters…
- Get a battery, test out battery life
- 3D print the enclosure, get out and write Forth on the go